1. Which of the four basic operations does the built-in structure array support?

*They do not support* insert *or* delete *operations since the size of the array cannot change. The only two operations it does support are* fetch *and* update.

2. Use the picture below to represent the memory allocated to a five-element, one-dimensional array of integers called *numbers*. Assume the first element of the array is stored at location 500, the array is stored in ascending memory locations, and that the integers occupy 4 bytes each. Label each element of the array. On the right side of the array indicate the memory location number (beginning at 500). Figure 2.1 in the text is an example of what this type of diagram should look like.

|  |  |  |
| --- | --- | --- |
|  | numbers | byte address |
|  |  | 500 |
|  |  | 504 |
|  |  | 508 |
|  |  | 512 |
|  |  | 516 |
|  |  |  |
| |  | | --- | |  | |  |  |
|  | 4 bytes |  |

3. Give the mapping function used to calculate the address of the *k*th element of a one-dimensional array of integers, assuming element zero is stored at location 500, the array is stored in ascending memory locations, and that an integer occupies 4 bytes of memory.

Address\_of\_node (N) = 500 + 4\*N

4.  20 integers are stored in a two-dimensional array of four rows and five columns. The name of the array is ages.

a) Label the rows and columns of the array as presented in the picture below, indicating the column numbers across the top of the array and the row numbers along the left side.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| 0 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  | x |  |
| 3 |  |  |  |  |  |

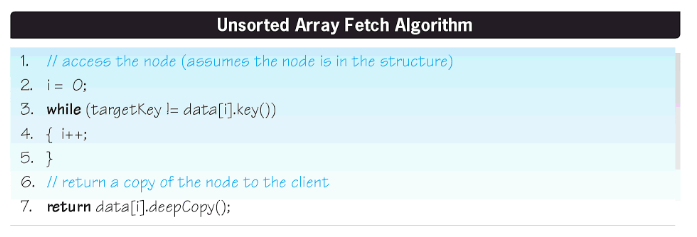
b) Place an x in the cell of the above picture whose contents is modified by the statement: ages[2][3] = 20;

c) Give the memory location of the cell ages[2][3] assuming the array is stored in row major order in ascending memory locations beginning at location 500. Assume each cell is 4 bytes.

N = 2\*(4 + 1) + 3 = 13

So that the address is: 4\*N+500 = 552

5. Give the change to **line 7** of the Unsorted Array structure’s Fetch algorithm shown below that would unencapsulate the structure after Fetch operation is completed.



**Note:** This is a single line of code replacing line 7.

*This seems like a strange question. Why would we want to unencapsulate the data?*

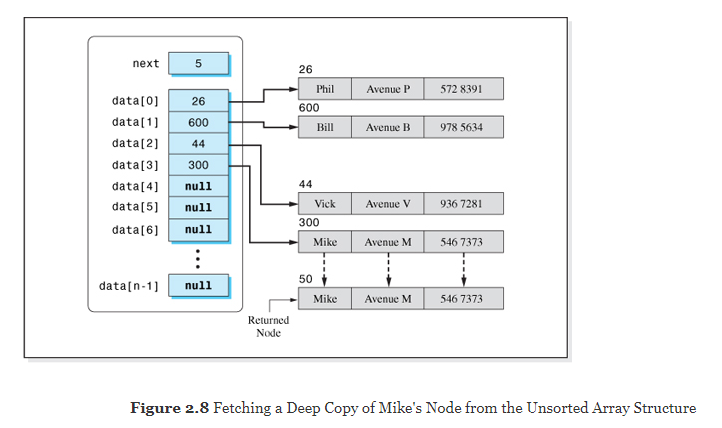
*Leaving out the deepcopy() would do the job. That way it returns an address of a location that should be private:* return data[i];

6. A data structure is to be chosen that will allow the customers to look up the price of an item, given its item number. The data set will only be loaded once and never modified. Of the three array-based structures discussed in this chapter, which would be best suited for this application?

*I would use a* sorted array *structure since fetch is the only required operation and it would generally be faster on a* sorted array, *assuming that all of the entries are equally likely. It does depend on the size of the data structure. Smaller structures may not see the optimization realized by the binary search.*

7. The Sorted Array structure accesses nodes using the Binary Search algorithm, and the Unsorted-Optimized Array structure uses a Sequential Search to access nodes. Knowing that a binary search is significantly faster than a sequential search, explain why it is that the average speed of the Unsorted-Optimized array structure is faster than that of the Sorted Array structure. (Assume all four basic operations are equally probable.)

*There are two features that could lead to this behavior. The first is that the fact that the Unsorted-Optimized Array could actually be faster on searches for the more probable nodes. The other feature is the fact that the search has to be done twice for each update operation in the binary search. Once for finding the node to be updated and once to put the new node into place.*  *If the array is large the logarithmic dependence of the speed function would mean the binary search would eventually dominate the effect of moving more probable searches up the list, even with having to do two searches in the lookup method.*

8.  Refer to the following data structure depicted in Figure 2.8 in the textbook:  


What memory location would be returned by the Fetch algorithm if Mike's node was requested and the structure was not encapsulated?

*If it's not encapsulated then the deepcopy won't be performed. That means that memory location 300 would be returned. The pointer to the location in the data structure.*

9. Consider the basic operation algorithms of the Unsorted and Unsorted-Optimized Array data structures.

1. What changes were made so that the Unsorted-Optimized Array structure performs better?

*The main difference is the way recently found nodes are moved up the structure. This makes the assumption that there are nodes that are more commonly referred to than others. It could actually be worse if we search for adjacent nodes in reverse order than they appear in the structure.*

1. Under what conditions would both structures' Fetch algorithms be equivalent from a speed viewpoint?

*It could happen if the nodes are searched in the order that they appear in the structure. Each one would get moved up but then would never be searched for again.*

*Another possibility is to do a large number of fetch operations making sure that every node is searched the same number of times in random order. There would be occasional attempts that are faster than average but they would be balanced by attempts that are slower than average. Each node would fluctuate around its initial location, sometimes moving up and sometimes moving down. On average, the fluctuations would cancel, because one moves down each time another one moves up. (One might expect that an RMS value for the fluctuations may not be zero but that won't change the overall timing.)*

10. The Unsorted-Optimized array structure is used to store a data set. Calculate its density if:

a) Each of the client's nodes contains 8 bytes of information and there are 50 nodes in the data set. *w=8, n=50.*

*Do this one both ways since n isn't large. First:*

*D= 1/(1+4/w+4/(nw) = 0.649*

*"rounded": D = 1/(1+4/w) = .667*

*The two results are significantly different.*

b) Each of the client's nodes contains 200 bytes of information and there are 1,000,000 nodes in the data set. *w=200, n=106.*

*The rounded formula can be used since n is large.*

*D = 1/(1+4/w) = 0.98.*

11. Give the average number of memory accesses of the Unsorted-Optimized array structure whose data set has 1,000,000 nodes and each client node contains 200 bytes:

a) Assuming all operations on the data set are equally probable.

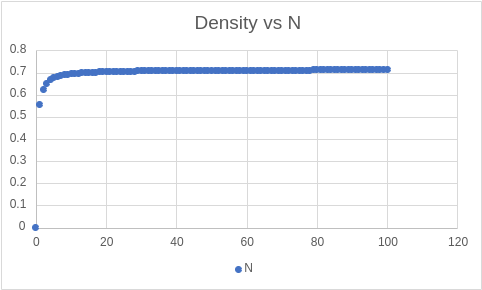
*This is an interesting question because there's nothing about it in the book. Each time we access a node we need to bring 200 bytes out of memory. That would be 200 memory accesses. The average number of nodes which have to be accessed is the same as the "speed factor" calculated in the book. So:*

*MA = (200)(5N+6)/4 = 2.5X108.*

b) Assuming only Insert operations will be performed on the data set.

*MA = (200)(3) = 600*

12. Plot the variation in density with the number of nodes, *n*, in an array-based structure. Assume each node contains 10 information bytes and the range is 2 ≤ *n* ≤ 100.   
  
**Note:** This question expects you to do a graph. An example of one of these density graph is given in the text as **Figure 2.14**. You can created a chart like this using Microsoft Excel with ***n*** in one column and the density in another column. Calculate the density using the formula **D = 1 / (1 + 4 / *w* + 4 / (w \* *n*))**, where ***w* =10**. Points for ***n*** should include (2, 10, 20, 30, 40, …, 90, 100). Use the X Y Scatter chart to create your graph. You then can copy and paste your resulting graph into this Word document here.



13. Define the data structure term *garbage collection*.

*Garbage collection refers to the process of removing blank nodes from the data structure after a delete operation. If this is not done the structure will fill up with empty nodes, wasting space.*

14. Describe the garbage collection method for the Unsorted-Optimized array structure.

*What is done is to move the last node in the structure into the newly emptied slot. This does not rearrange the pointer table. The new address is stored where the old one was.*

15. You have coded an application for your friend's business that uses an Unsorted-Optimized array to store the venture's data. One day your friend informs you that the speed of the operations performed on the data set seems to be getting faster and faster. Explain how this could happen.

*The Unsorted-Optimized array has that feature where it moves data up the table after a lookup. This puts the most-probable nodes near the top of the list. It is likely that his business has certain nodes that are referenced on a regular basis while others are rarely used. Since the most likely nodes are the first ones searched, they are also found relatively quickly.*

16. An application's data set will consist of five different types of nodes. Is the data set homogeneous or heterogeneous?

*The data set is heterogeneous.*